Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
MAR 2006		**00-12-2005 to 00-03-2006**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
Irreducible Truths of Software-Intensive Program Management	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Defense Finance Accounting Service,Indianapolis,IN,46205**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES
Defense AR Journal, December 2005-March 2006

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			
unclassified	**unclassified**	**unclassified**	**Same as Report (SAR)**	**12**	

IRREDUCIBLE TRUTHS OF SOFTWARE-INTENSIVE PROGRAM MANAGEMENT

David Cottengim

This article argues that the odds are against a software-intensive program achieving the goals and objectives established in the initial acquisition program baseline. By most objective measures of success—cost, schedule, and performance baselines—almost every software-intensive program proves unsuccessful. Reasons for these failures are examined and solutions for improvement are discussed.

For decades the acquisition community has applied standard engineering and scientific principles to improve the professions of software engineering and program management. The lessons learned from this process have been distilled into a few succinct, commonly held beliefs. Some of these maxims, applied to software-intensive program management, may sound familiar:

1. Adding manpower to a late software project makes it later (Brooks & Fredrich, 1975),

2. Hope is not a strategy (Page, 2003), and

3. Real programmers don't need sleep (Yourdon, 1997, p. 61).

To focus the theme of this exposition, one more should be added:

4. Almost all software-intensive programs fail.

FAILURE DEFINED

To further investigate maxim #4, a common definition of failure is needed. By statute and regulation, if the estimated costs of a program increase 15 percent above what was

in the latest approved Acquisition Program Baseline (APB), or if any milestone from the latest approved APB schedule is delayed by 6 months or more, then the program is in breach (U.S.C. Title 10, §2432 and §2433). One need merely replace the phrase "latest approved APB" with "first approved APB" to illuminate the definition of failure to be used in this article.

The contention of this article is simple: the odds are overwhelmingly against a software-intensive program achieving the goals and objectives established in its initial APB. By any objective measure of success, almost every software-intensive program is probably going to deviate substantially from its initial cost, schedule, and performance baselines. For most readers, this is not considered news. However, it is important that we all begin from this common understanding.

CURRENT STATE OF SOFTWARE-INTENSIVE PROGRAM MANAGEMENT

As one author wrote, managing large projects is a curse you should avoid (Lientz, 1999, p. 250). Even after years of improvement in program management and software engineering tools and techniques, the profession is still plagued by embarrassing failures. In a recent study of 250 software-intensive projects with 10,000 or more function points, only about 25 were deemed successful (i.e., achieved their initial cost, schedule, and performance objectives) (Jones, 2004, p. 5).

The same study revealed that the following six factors were common in the 25 successful programs:

1. Project planning: "Planning is the answering of the following questions: What must be done? How should it be done? Who will do it? By when must it be done? How much will it cost? How good does it have to be?" (Lewis, 2000, p. 49).

2. Cost estimating: "Cost estimating and analysis is that portion of systems analysis dealing with the tasks of conceptual modeling, output and cost measurement, verification cost and output prediction, and evaluation and comparison of the costs of each alternative" (Society of Cost and Estimating, 2005).

3. Measurement techniques: If you do not measure a thing you cannot control it. "Measurements are the basis for detecting deviations from acceptable performance" (Florac, 1999, p. 7).

4. Milestone tracking: "A milestone is an intermediate objective that defines an important, measurable event in the project and represents a result that must be achieved at that point. Clearly defined milestones are essential for monitoring progress, especially in large and/or long-term projects" (Kerzner, 2000, p. 82).

5. Change management: "Change management is an important responsibility of any acquisition program" (Space & Missile Systems, 2004). Change management must

be applied to all program deliverables (e.g., software, manuals, training materials, planning documents, requirements).

6. Quality control: Quality control refers to "the operational techniques and activities used to fulfill requirements for quality" ("Definitions," 2005).

While the effective implementation of these six factors may be inherent to successful programs, they alone are not sufficient to guarantee program success. There are eight common reasons cited for the proliferation of program failures (Posner, 1987):

1. Inadequate resources,
2. Unrealistic schedules,
3. Unclear goals and senior executive direction,
4. Uncommitted team members,
5. Inadequate planning,
6. Communication breakdowns,
7. Goal and resource changes, and
8. Interdepartmental conflicts.

The General Accounting Office (GAO) recently reported that a 1999 study performed by the Standish Group found that about one-third of software development programs resulted in cancellation. Furthermore, in a series of studies completed through the 1990s, the average cost overrun was 189 percent, while the average schedule overrun was 222 percent of the original estimate. On average, only 61 percent of the projects were delivered with originally specified features or functions (General Accounting Office, 2004, p. 9).

ATTEMPTS TO FIX THE PROBLEM

To redress this reality there has been a perennial effort to improve the education and training of would-be and existing program managers (PMs) in the specific skills of program management and software engineering and to implement quality improvement programs wherever possible. Professional organizations such as the Project Management Institute (PMI) and industry-accepted publications such as PMI's *A Guide To The Project Management Body of Knowledge* now exist to facilitate making program management a standardized profession. A subculture of acquisition reform has evolved within DoD (Cho, Jerrell & Landay, 2000). Defense has been at the forefront

of institutionalizing sound program management principles in its training requirements for current and future program managers (Department of Defense, 2002).

Congress is even helping by mandating implementation of software acquisition process improvement programs. The National Defense Authorization Act of 2003 states, "The Secretary of each military department shall establish a program to improve the software acquisition processes of that military department" (section 804, Public Law 107-314). The DoD policy, as expressed in a February 20, 2004, policy memo issued by then Under Secretary of Defense for Acquisition, Technology and Logistics Michael W. Wynne, has reiterated the need for a more formal, structured, and robust systems engineering approach to all programs regardless of acquisition category.

Congress is even helping by mandating implementation of software acquisition process improvement programs.

These overt efforts to create and require formalized program management training, revise acquisition guidelines, implement sound program management techniques, require software engineering plans, and improve the software acquisition process are admirable, yet these efforts alone will be insufficient. Poor program management techniques, sloppy software engineering, and non-repeatable software development processes are not the proximate cause of most major software program failures. The root cause of the eight reasons for failure listed above is not a lack of quantitative program management or software engineering skills. Nor is the continued use of immature and unstructured software development processes to blame. Post-project analysis has taught us that failures are more behavioral than quantitative (Kerzner, 2000, p. 175). If we are to solve the problem of program failures we must first accurately diagnose its genesis.

SUB-OPTIMIZATION

One might characterize highly complex acquisition environments as sub-optimization engines. Complex acquisition environments are structured to produce sub-optimal results (presumably an unintentional situation). It is virtually impossible to produce optimal solutions in an environment that is characterized by:

- Frequently changing strategic directions,
- Turnover in senior leadership,
- Rapidly changing technology,

- Entrenched legacy business practices,
- Freelance development outside the control of the standardization efforts,
- Massive documentation requirements disconnected from actual program realities, and
- Confusing and conflicting budgeting and contracting requirements and procedures.

These are all known problems. These are areas of program management and software engineering which receive extensive coverage in professional training courses. Regrettably, even when all these known environmental complexities are diligently addressed, we still see programs either fall short of initial expectations or fail.

With so many preventive measures being taken to ensure success, why then do software-intensive programs continue to produce sub-optimal solutions?

THE ACCOUNTABILITY MYTH

For every PM there will come a time when the inevitable conclusion is reached: nobody appears to be in charge of whatever mission your program is trying to support. Some acquisition organizations have tried to redress this lack of a controlling authority by assigning specific accountability to a PM for success. The mythology of accountability suggests that by making program managers accountable for outcomes, they are more likely to take the actions necessary for the program to succeed. The reality is that accountability does not produce success. Accountability often merely provides a convenient scapegoat for the inevitable failure produced by environments that are toxic to success.

Uncertainty over authority coupled with certainty about who is accountable for failure causes conflict between PMs and stakeholders. How many times have program managers been in a meeting where the conclusion was that "X" needed to happen? Then everyone looked at each other and concluded that they knew of no one who could make "X" happen. And if such a person did exist, it certainly was not the PM. This type of situation has been called a "locus-of-authority conflict," which is produced naturally in matrix organizations (Meredith & Mantel, 2000, p. 238).

COLLECTIVELY IMPOSSIBLE DEMANDS

These locus-of-authority conflicts (i.e., nobody is in charge and every stakeholder wants their individual requirements satisfied) lead to another postulate of software-intensive program management:

In complex acquisition environments, program managers must foster cooperation among equals.

No amount of training in the technical skills of program management will overcome the simple truth that, as a PM, you cannot make people do what you need them to do. This is the root cause of many software-intensive program failures. Stakeholders often cannot agree on priorities, refuse to standardize business practices, take off on their own proprietary solutions, or simply refuse to participate in the program.

It should be no surprise that few software-intensive programs ever achieve the lofty objectives set forth in their initial APB. A reasoned analysis of our current acquisition environment would rationally conclude that most expectations of contemporary programs are unrealistic. The cruel reality is that we train PMs and then drop them in an organizational "shark tank" that opposes many of the principles they have just absorbed in their training. Program managers often find themselves in a superfluous role, accountable yet powerless.

THE PROGRAM MANAGER'S DILEMMA

Game theory tells us we can presume that decision-makers are rational. They are aware of their alternatives, form expectations about any unknowns, have clear preferences, and choose their actions deliberately after some process of optimization (Osborne & Rubinstein, 1994, p. 4). So why do stakeholders often take actions that undermine the objectives of approved programs? The answer is found in a simple principle: "It is impossible to maximize two or more functions simultaneously" (Von Neumann & Morgenstern, 1944, p. 11). This phenomenon has been labeled "the program manager's dilemma" (Ward, 2004, p. 54). When stakeholders believe that their goals are incongruent with the goals of the program (the presumed optimal solution), they will rationally choose to not cooperate (Bartol & Martin, 1991, p. 61). Stakeholders often believe that they cannot simultaneously do what is best for themselves and the approved program. Changing such beliefs and the resultant behaviors will do more for program success than any level of training or acquisition reform.

THE SOLUTION

How can the behavior of stakeholders be changed so that they will support rather than oppose the efforts of a program manager? One might think that the logical approach would be to change the beliefs that give rise to the undermining behavior, but that is a losing strategy. Beliefs are difficult to measure. Behavior on the other hand is observable and measurable.

Therefore the solution must focus on stakeholder behavior rather than the core belief systems that cause behavior (Beer, Spector, & Eisenstat, 1990). Behavior can be observed and rewarded or discouraged as appropriate. Once new behaviors are adopted, core beliefs and values will be influenced by the success the new behavior fosters. Thus focusing on behavior indirectly influences the core belief systems that need to be affected.

Acceptable behavior (that which advances the objectives of approved programs) within the organization needs to be encouraged and unacceptable behavior (that which undermines the objectives of approved programs) needs to be discouraged. If the punishment for not cooperating is so great that cooperation is the best choice in the short run, then there is no longer a dilemma (Axelrod, 1984). This link between behavior and performance has been recognized as part of the proposed National Security Personnel System (NSPS): "The NSPS regulations provide for consideration of employee behavior as a performance factor, element, or objective, such as 'teamwork/cooperation' (Federal Register, 2005, p. 7562).

CONCLUSION

The technical, social, and engineering skills required to be a better program manager or software engineer are necessary but not sufficient for success with a software-intensive program. The organizational environment *must* support the program objectives or the program will most likely fail.

The proposed solution presented above presupposes someone in power to observe behavior and reward or punish stakeholders as appropriate. A PM must identify the senior leaders who have the greatest potential to impact the ongoing viability of the program (Pinto, 1996) and obtain their support. This more than any other factor will determine success or failure for most programs. A PM's ability to find such support is more critical than any technique, skill, or tool taught in the classroom.

Obtaining senior leadership support is not a revolutionary concept. Think for a moment about all the quality or process improvement techniques to which you have been exposed. Improvement programs all have the same prerequisites for success. They all require something like:

- Management commitment (Wood & Silver, 1995, p. 166),
- Commitment to perform (Paulk, 1994, p. 47),
- Senior management leadership (Humphrey, 1989, p. 19), and
- Commitment to change; senior leadership must understand and be completely behind implementing the change (Harry & Schroeder, 2000, p. 281).

Just about any process can work if it has real management support. Process cannot make up for lack of leadership support, but leadership support can mitigate process deficiencies. Do not forget that the success of the program ultimately depends on the ability to get stakeholders to do the things they often do not want to do. The behavior of individuals acting for or against the mission objectives controls the chance for success more than any other professional tool or skill employed on an acquisition program.

David Cottengim is an Accountant at the Defense Finance and Accounting Service, Indianapolis. He has over 14 years experience in system development for Department of Defense activities. He is certified as a Project Management Professional, Certified Software Test Engineer, Certified Treasury Professional, and Certified Government Financial Manager. He completed his undergraduate education in Finance and Economics and his graduate education in Finance and Management Information Systems at the Indiana University Kelly School of Business.

E-mail address: DAVID.COTTENGIM@DFAS.MIL

AUTHOR **BIOGRAPHY**

REFERENCES

Axelrod, R. (1984). *The evolution of cooperation*. New York: Harper Collins.

Bartol, K. M., & Martin, D. C. (1991). *Management*. New York: McGraw-Hill.

Beer, M., Spector, B., & Eisenstat, R. (1990, October). *Critical path to corporate renewal*. Boston: Harvard Business School.

Brooks, J. R., & Fredrich P. (1975). *The mythical man-month*. Reading, MA: Addison Wesley.

Cho, G., Jerrell, H., & Landay, W. (2000, January). *Program management 2000: Know the way, how knowledge management can improve DoD acquisition*. Ft. Belvoir, VA: Defense Systems Management College Press.

Definitions of Quality Assurance/Quality Control. (2005). American Society for Quality website. Retrieved April 25, 2005, from http://www.asq.org/topics/qa_qc.html

Department of Defense. (2002, March). *Acquisition career development program*. (DoD Manual 5000.52-M).

Federal Register. (2005, February 14). National security personnel system, *70*(29), 7562.

Florac, W. A. (1999). *Measuring the software process, statistical process control for software process improvement*. Reading, MA: Addison-Wesley.

General Accounting Office. (2004, March). *Defense acquisitions, stronger management practices are needed to improve DoD's software-intensive weapon acquisitions* (publication no. GAO-04-393). Washington, D.C.: author.

Harry, M., & Schroeder, R. (2000). *Six sigma*. New York: Doubleday.

Humphrey, W. S. (1989). *Managing the software process*. Reading, MA: Addison-Wesley.

Jones, C. (2004, October). Software project management practices: Failure versus success. *Cross Talk*, *17*(10), 5–9.

Kerzner, H. (2000). *Applied project management*. New York: John Wiley & Sons.

Lewis, J. P. (2000). *The project manager's desk reference*. New York: McGraw-Hill.

Lientz, B. P., & Rea, K. P. (1999). *Guide to successful project management*. New York: Harcourt Brace.

Meredith, J. R., & Mantel, S. J. (2000). *Project management: A managerial approach* (4th ed.). New York: John Wiley & Sons.

National Defense Authorization Act of 2003. (2003). Public Law 107-314, Section 804.

Osborne, M. J., & Rubinstein, A. (1994). *A course in game theory*. Cambridge, MA: MIT Press.

Page, R. (2003). *Hope is not a strategy*. New York: McGraw-Hill.

Paulk, M. C. (1994). *The capability maturity model: Guidelines for improving the software process*. Reading, MA: Addison-Wesley.

Pinto, J. K. (1996). *Power & politics in project management*. Upper Darby, PA: PMI Publications.

Posner, B. Z. (1987, March). What it takes to be a good project manager. *Project management journal*, *18*(1), 51–54.

Project Management Institute. (2004). *A guide to the project management body of knowledge* (PMBOK® Guide) (3rd ed.). Newton Square, PA: Author.

Space & Missile Systems Center, U.S. Air Force. (2004, January). *Systems engineering primer and handbook*.

The Society of Cost and Estimating Analysis. (2005). *The body of knowledge for cost estimating and analysis*. Retrieved April 22, 2005, from http://www.sceaonline.net/content.asp?contentid=205

Von Neumann, J., & Morgenstern, O. (1944). *Theory of games and economic behavior*. Princeton, NJ: Princeton University Press.

Ward, D. (2004, May-June). The program manager's dilemma. *Defense AT&L*, *33*(3), 54–57.

Wood, J., & Silver, D. (1995). *Joint application development*. New York: John Wiley & Sons.

Yourdon, E. (1997). *Death march*. Upper Saddle River, NJ: Prentice Hall.